A Machine Learning-based Framework to Support Monoliths to Microservices Migration: Identification, Packaging, and Deployment

Imen Trabelsi

École de technologie supérieure ÉTS, Montréal, Canada

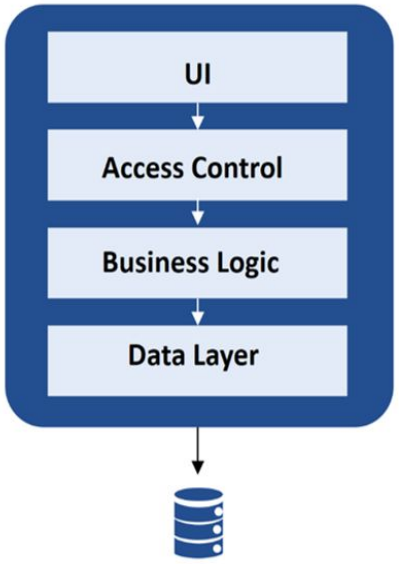
Supervised by

Prof. Naouel Moha

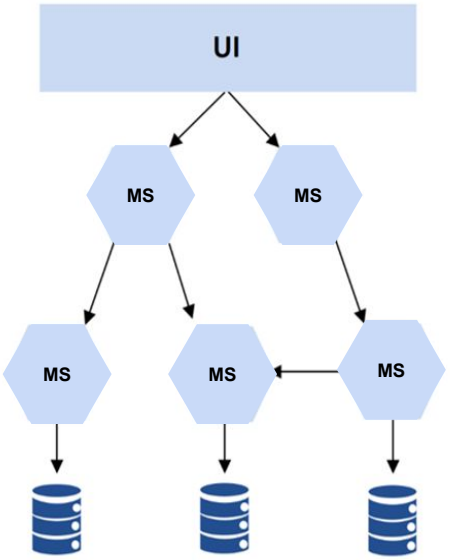
Prof. Yann-Gaël Guéhéneuc

18 Aug 2025

Monoliths vs. Microservices



vs.



Monolith

Microservices

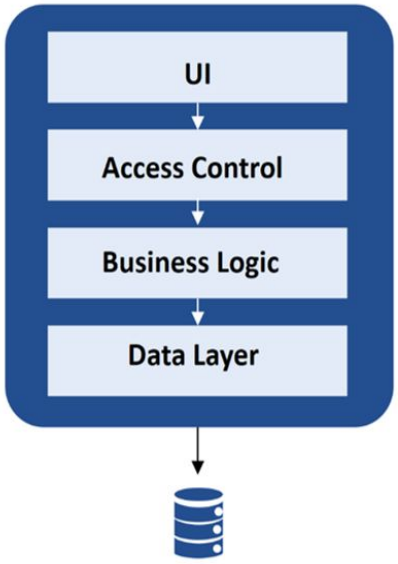
Monoliths to Microservices

High Maintenance Cost

Lack of Flexibility

Lack of Scalability

Lack of Technical Support

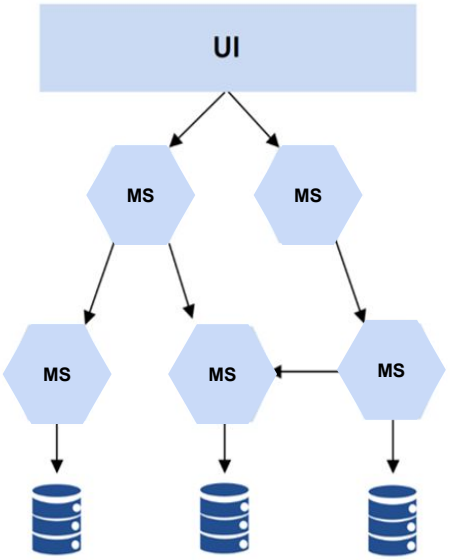


Monolith

amazon ebay

Migration

NETFLIX Spotify



Microservices

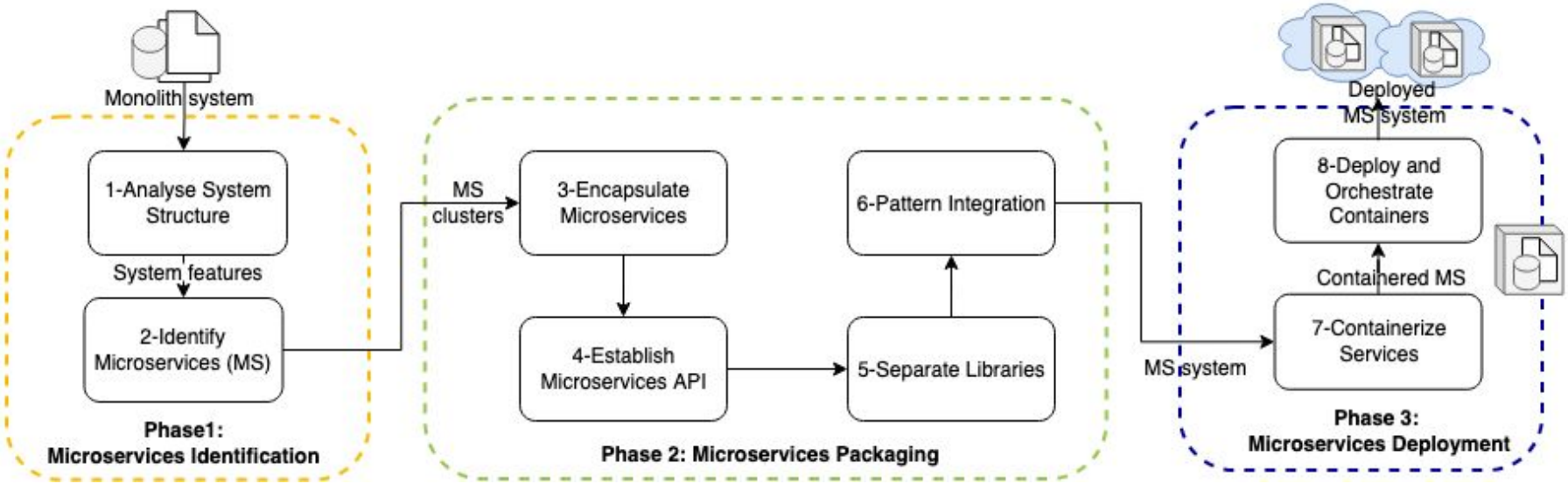
Lower Maintenance Cost

Accelerate Time-to-market

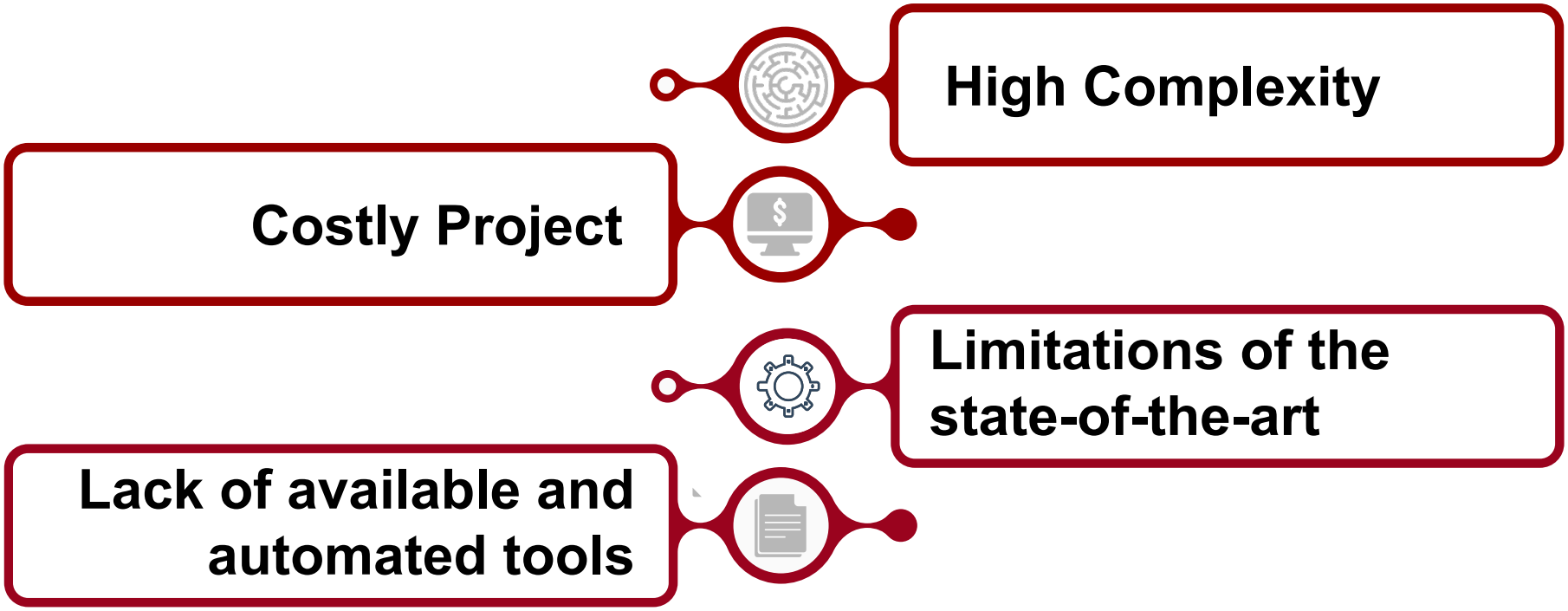
Scalable Application

Increase Availability

Migration Process



Migration Challenges

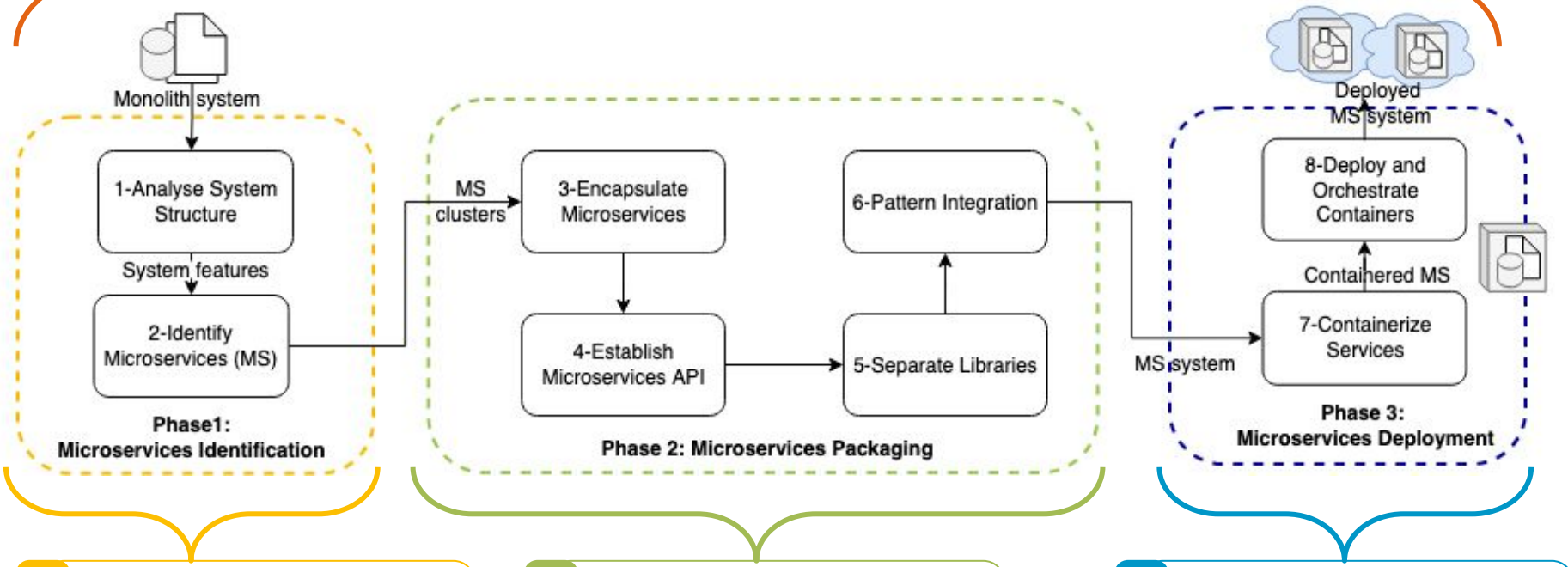


Thesis Premise

- Machine learning excels at uncovering patterns and structural relationships in source code
- Large language models demonstrate strong capabilities in understanding, generating, and transforming code

Thesis Problems

P1 Limited Understanding of ML-Based Migration Approaches



P2 Lack of Automated and Domain-aware Identification Approaches

P3 Absence of Automation in the Packaging Phase

P4 Manual and Template-based Deployment Configuration

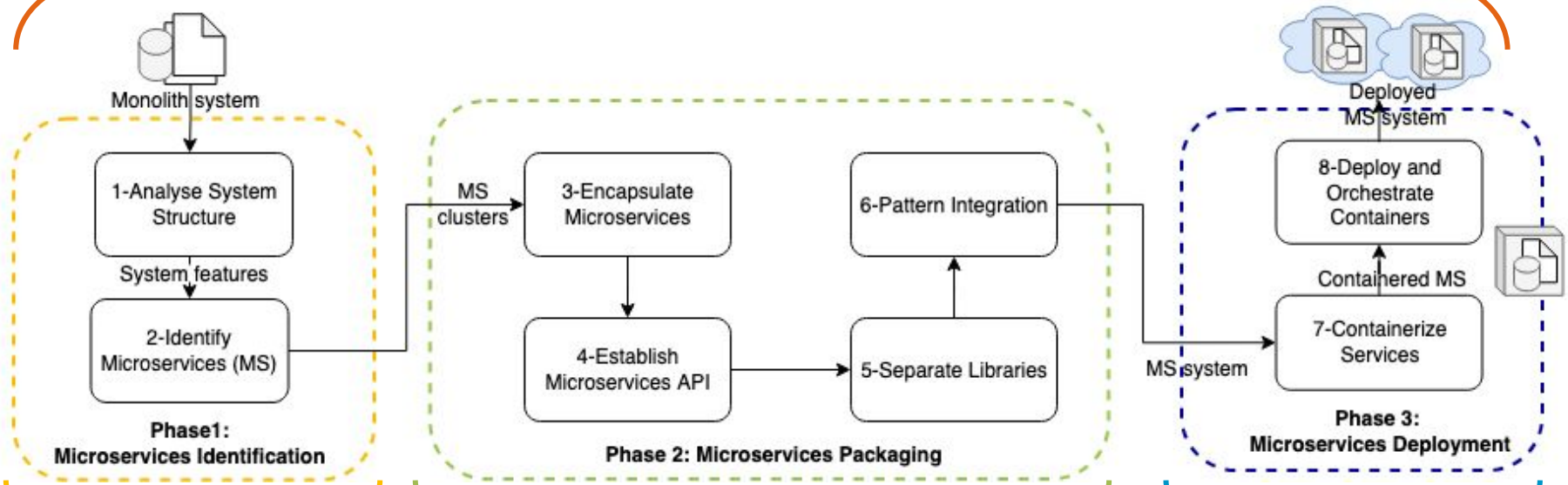
Lack of Automation

Thesis Statement

Can monolith to microservices migration be fully automated, while ensuring accuracy and minimising manual effort, by leveraging machine learning and large language models across the phases of service identification, packaging, and deployment.

Thesis Contributions

C1 A SLR of ML-based Migration Approaches



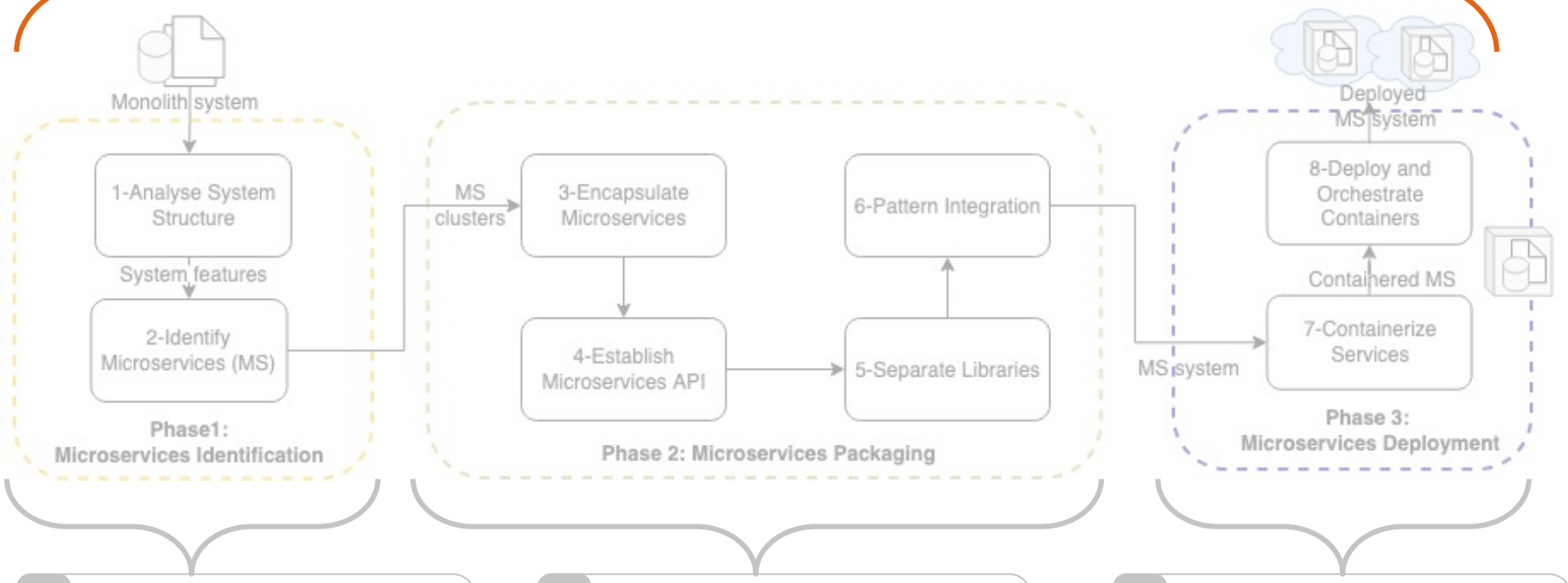
C2 MicroMiner: ML-based, Automated Identification Approach

C3 MicroPacker: LLM-based, Automated Packaging Approach

C4 MiDKo: LLM-based, Fully Automated Deployment Manifests Generation

Engineering Research

C1 A SLR of ML-based Migration Approaches



C2 MicroMiner: ML-based, fully automated identification approach

C3 MicroPacker: LLM-based, automated packaging approach

C4 MiDKo: LLM-based, Fully automated deployment manifests generation

SLR of ML-based Migration Approaches

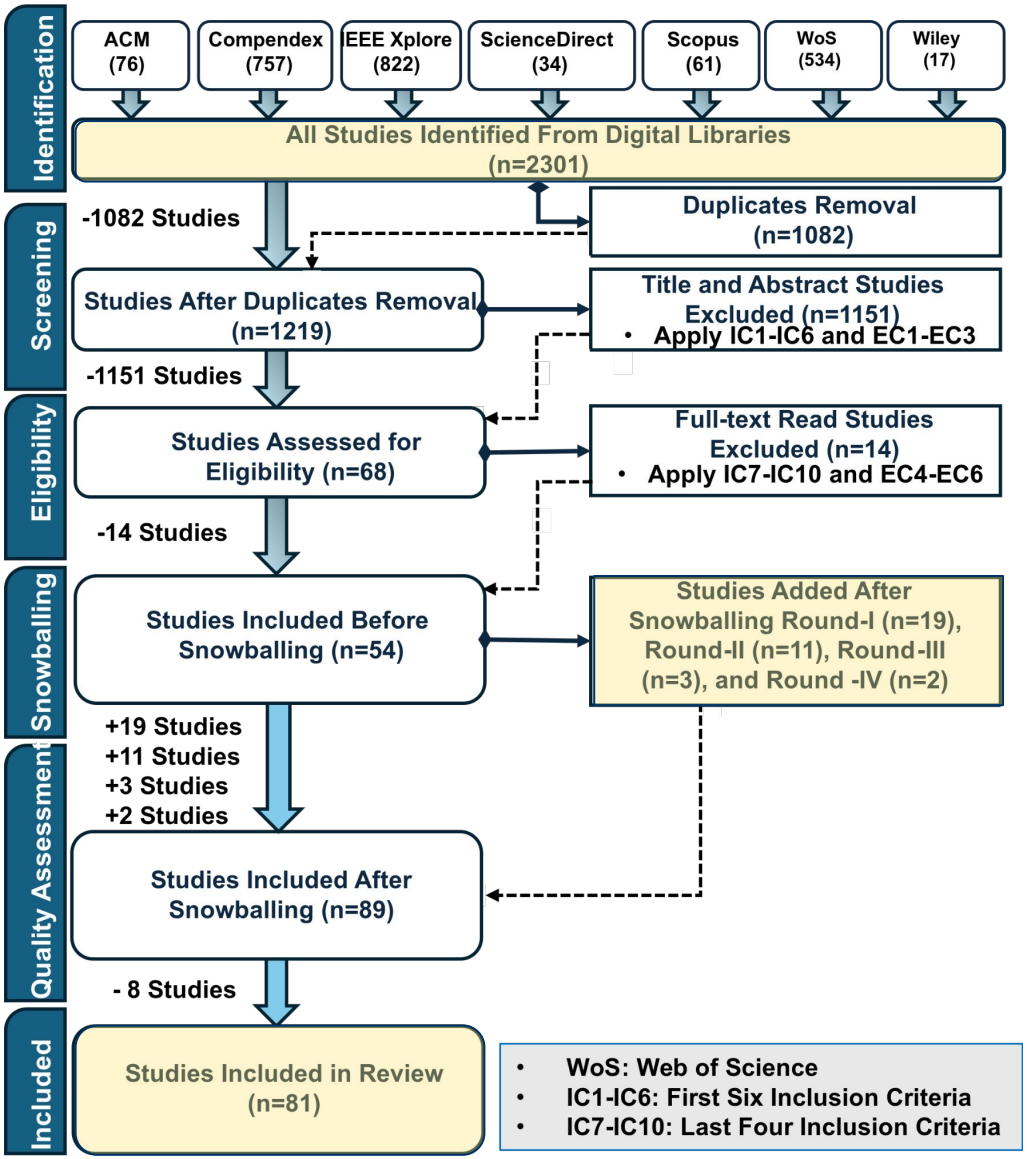
RQ1: Which migration **phases** are automated by ML?

RQ2: How are **inputs** used by ML migration approaches?

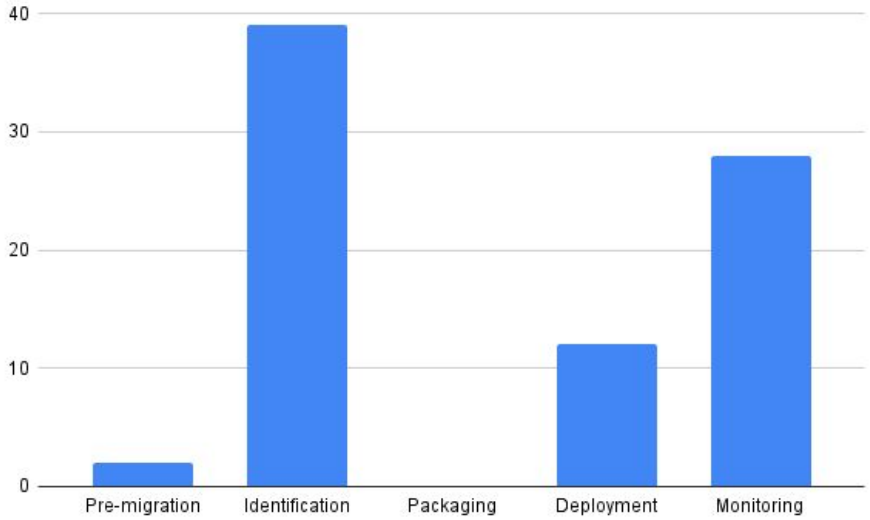
RQ3: What **ML approaches** do researchers apply?

RQ4: How do researchers **evaluate** ML approaches?

PRISMA Flow for Studies Selection

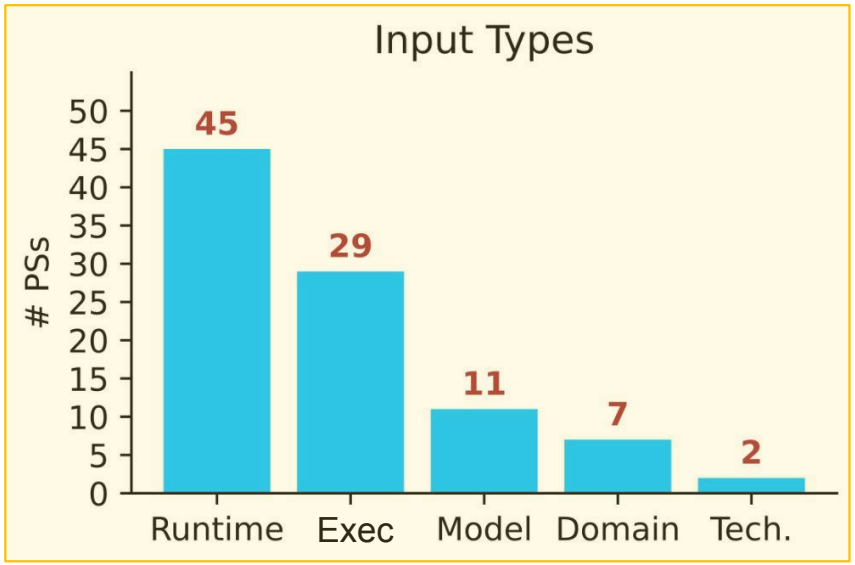


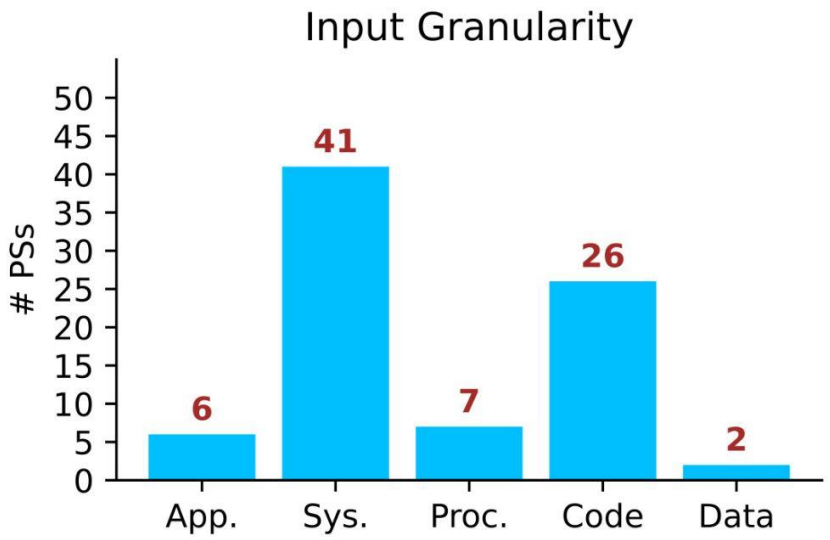
Migration Phases Automated by ML (RQ1)

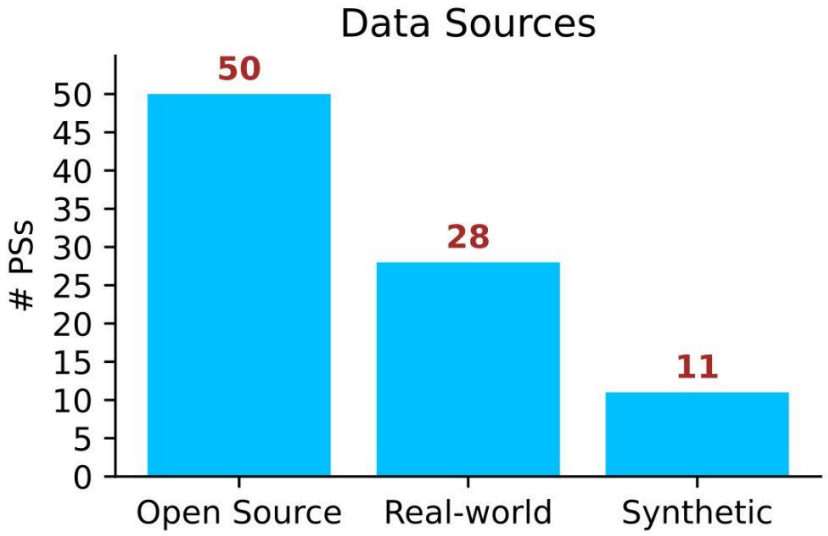


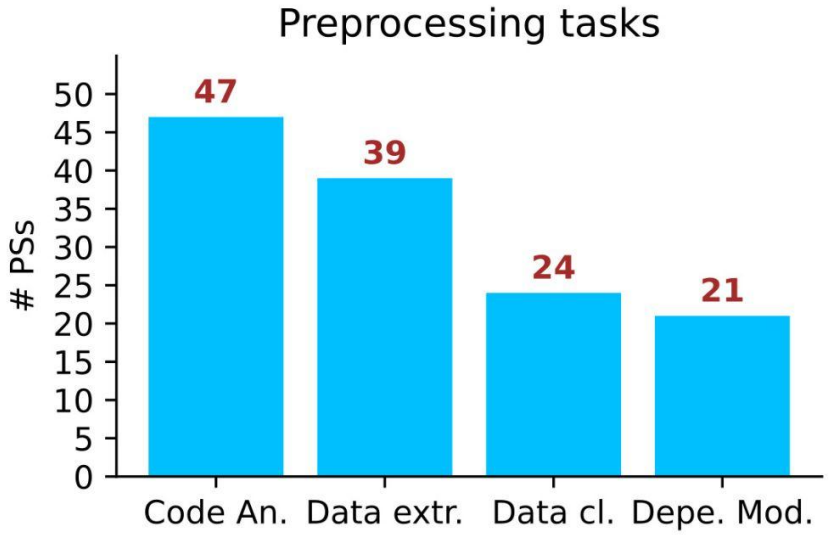
- Identification is the most studied phase (39), followed by Monitoring (28), Deployment (12), and Pre-migration (2)
- **Packaging** is completely **unaddressed** by current ML approaches

Inputs Used by ML Migration Approaches (RQ2)

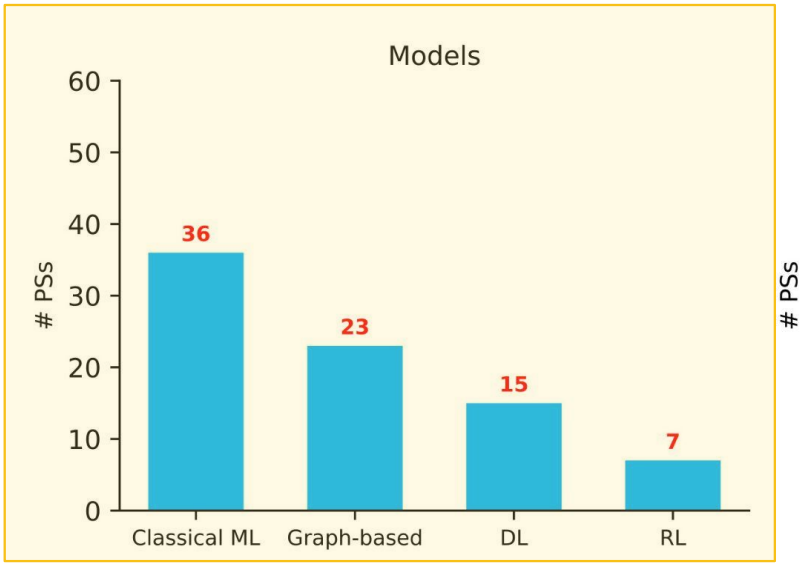


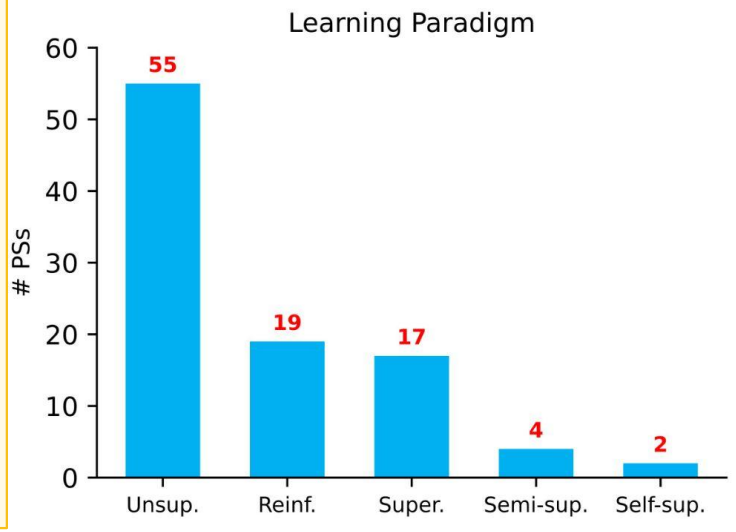


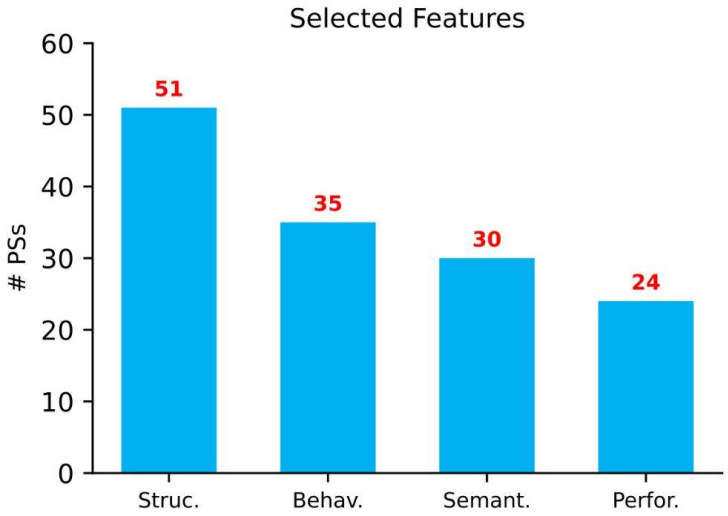




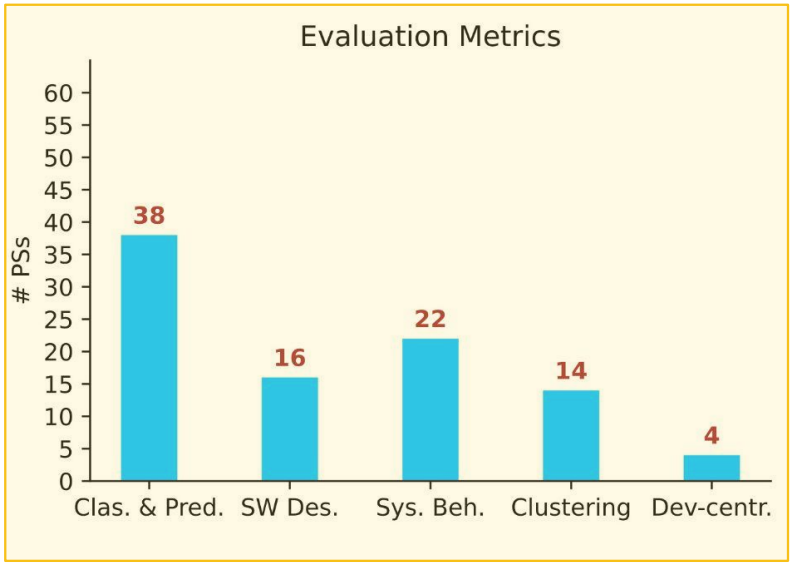
ML Approaches Used During Migration (RQ3)

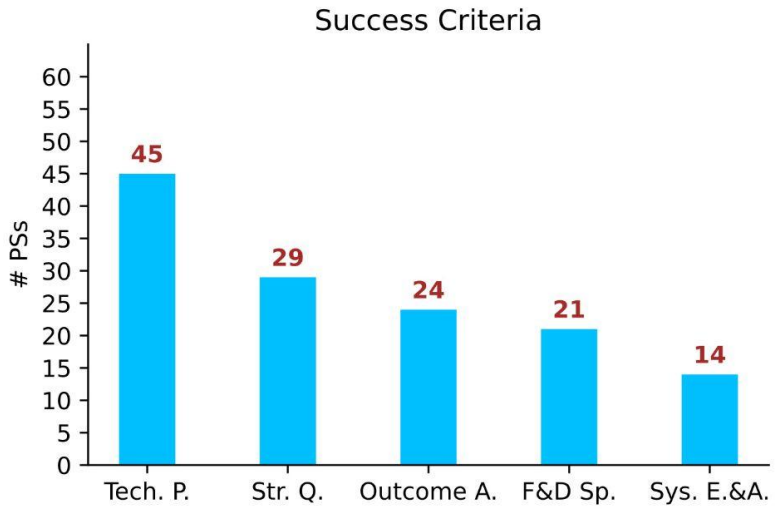


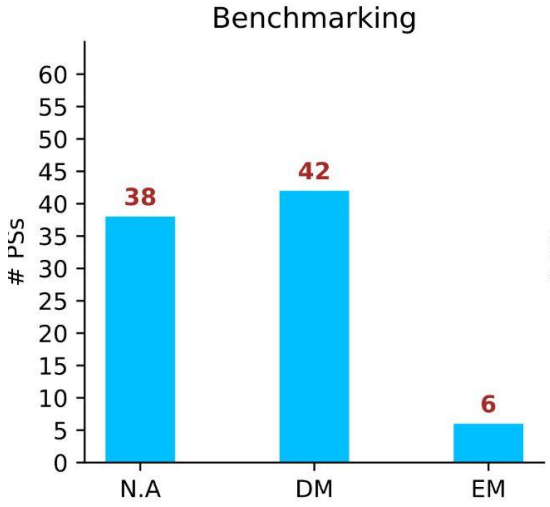


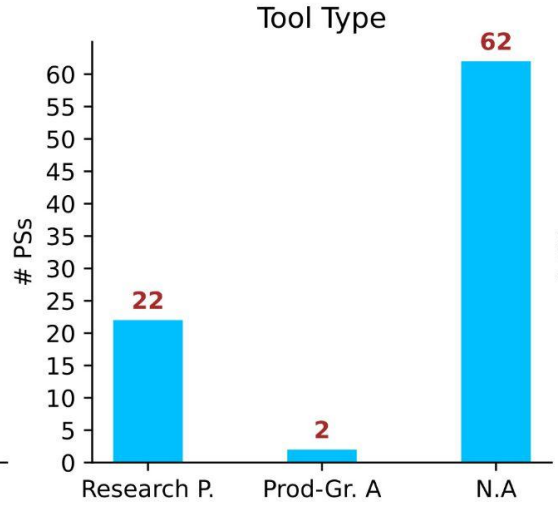


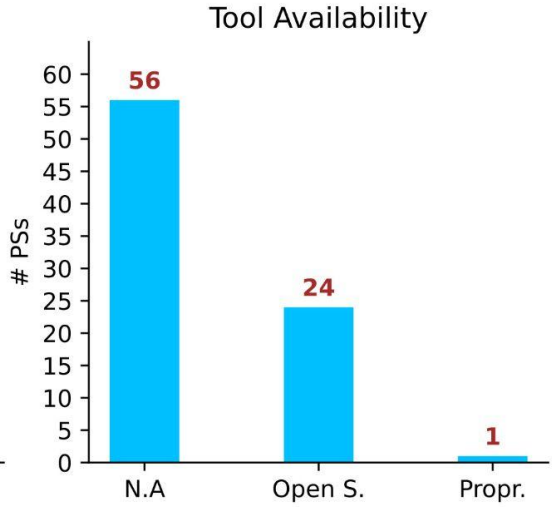
Evaluation of ML Approaches (RQ4)











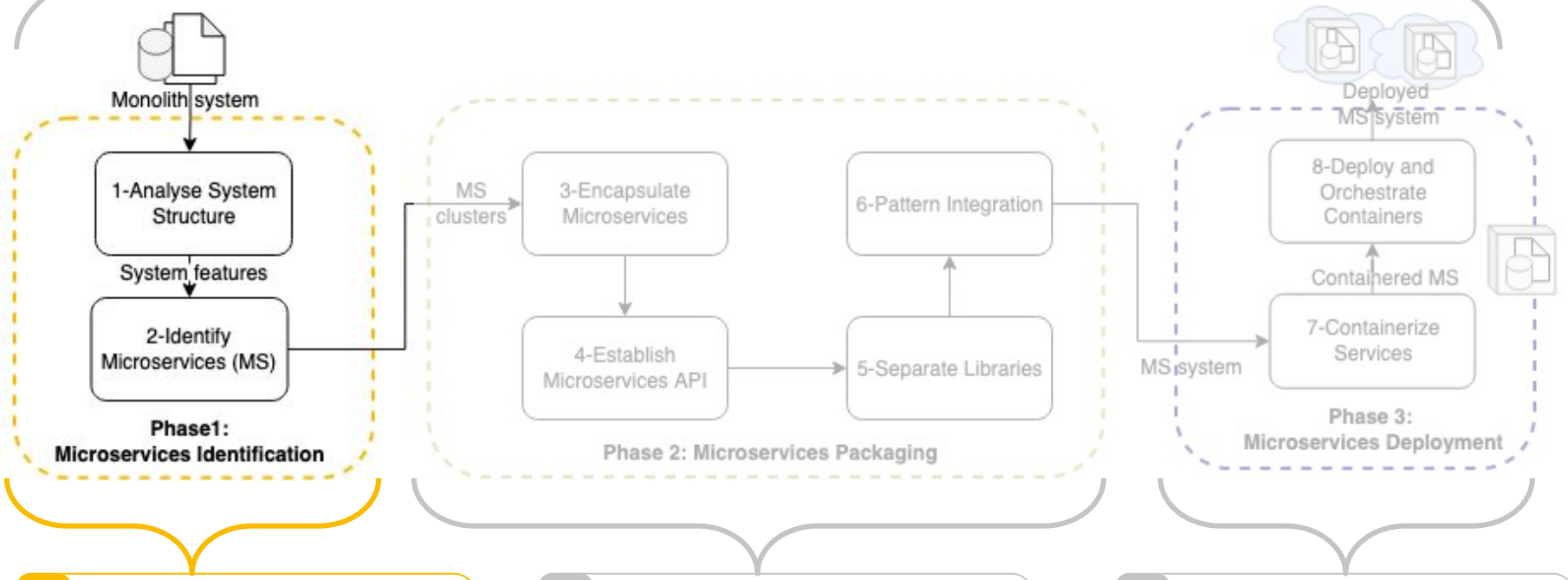
Research Gaps

1. Over-reliance on structural metrics
2. Lack of automation in the packaging phase
3. Overlooked deployment artifact generation
4. Limited tool accessibility and automation
5. Insufficient dataset availability and diversity
6. Absence of standard evaluation frameworks

Research Gaps

1. Over-reliance on structural metrics
2. Lack of automation in the packaging phase
3. Overlooked deployment artifact generation
4. Limited tool accessibility and automation
5. Insufficient dataset availability and diversity
6. Absence of standard evaluation frameworks

C1 A SLR of ML-based migration approaches



C2 MicroMiner: ML-based, Fully Automated Identification Approach

C3 MicroPacker: LLM-based, automated packaging approach

C4 MiDKo: LLM-based, Fully automated deployment manifests generation

Our Contributions for Identification Phase

- MicroMiner [2] : A type-based Approach for Microservices Identification using Machine Learning and Semantic Analysis
- MicroMatic [3]: Fully Automated Microservices Identification Approach From Monolithic Systems
- MAGNET [4]: Method-based Approach using Graph Neural Network for Microservices Identification

Our Contributions for Identification Phase

- MicroMiner [2] : A type-based Approach for Microservices Identification using Machine Learning and Semantic Analysis
- MicroMatic [3]: Fully Automated Microservices Identification Approach From Monolithic Systems
- MAGNET [4]: Method-based Approach using Graph Neural Network for Microservices Identification

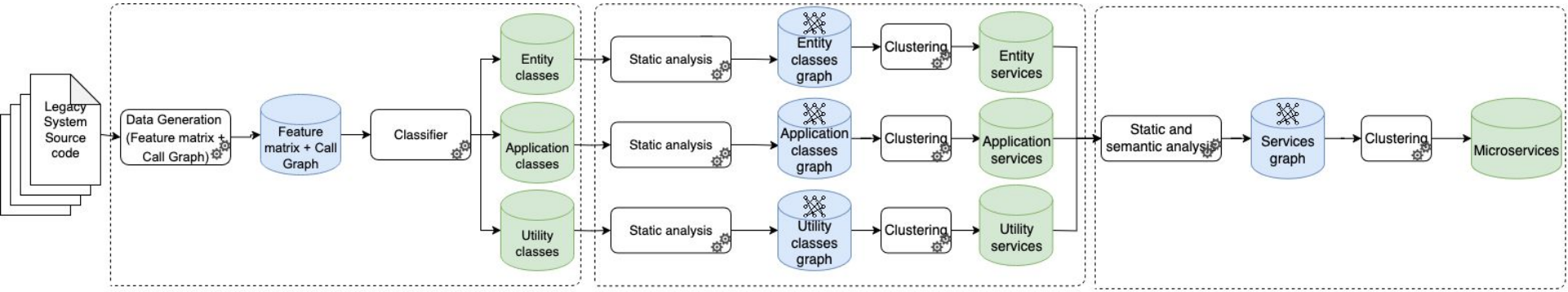
Context

- Microservice identification is recognized as the **most difficult** phase of migration
- Must ensure alignment with domain concepts like **bounded context**
- Most rely solely on **structural metrics**

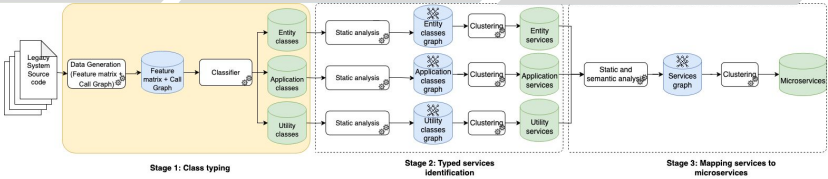
  (e.g., cohesion, coupling)
- Incomplete **architectural layers** within individual microservices
- Lack of **automation** and tooling

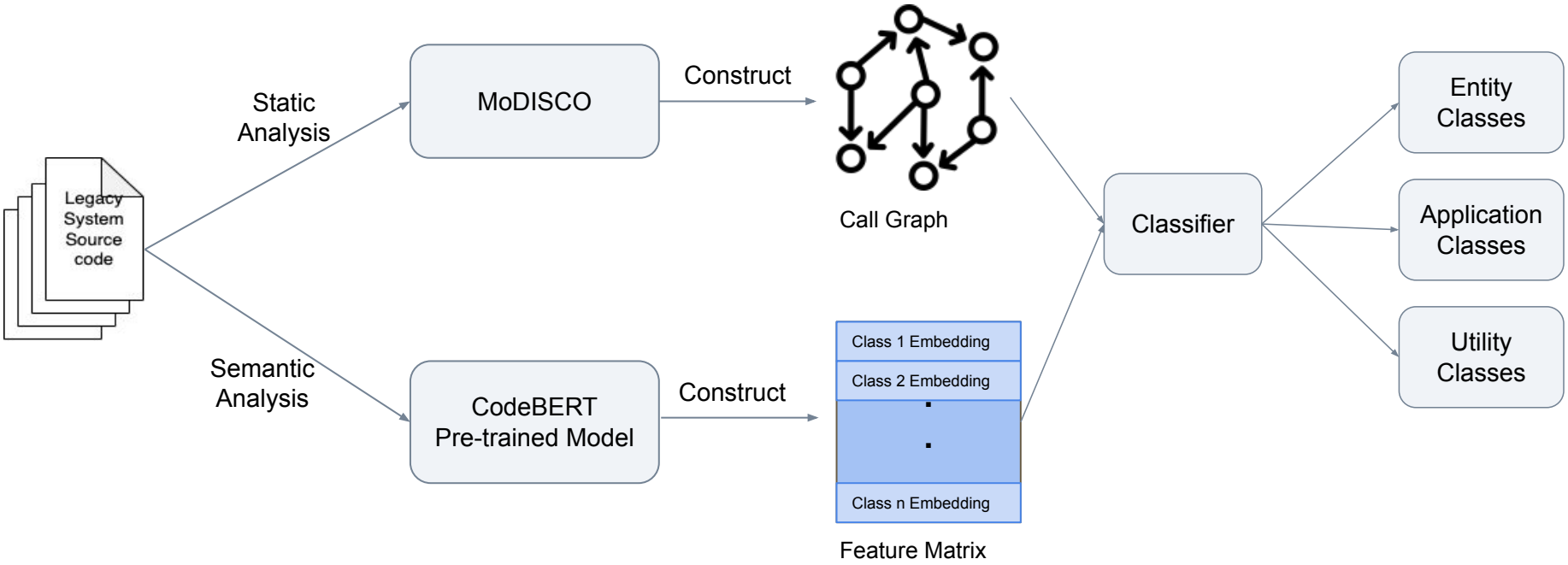
MicroMiner Overview

- ✔ Based on semantic and static analyses
  - Respecting bounded context principle
- ✔ Based on typed services
  - Insuring complete architecture layers
- ✔ Provides an automated tool

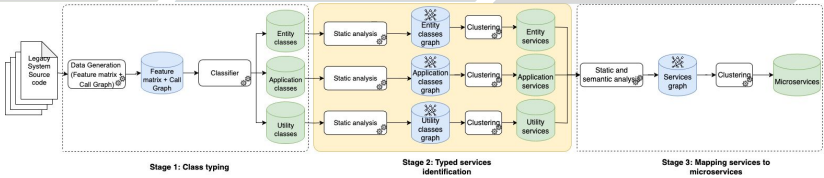


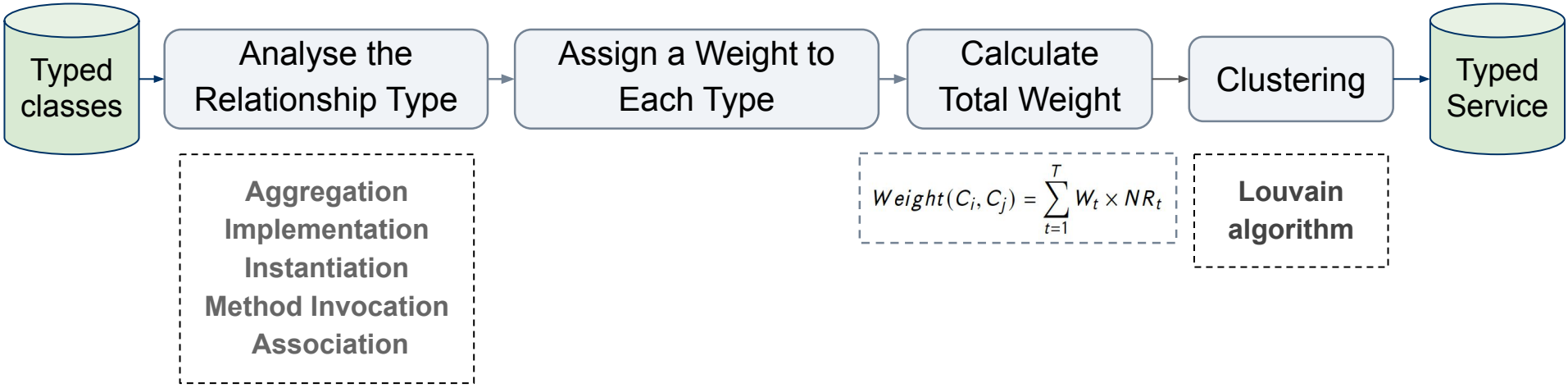
Stage 1: Class Typing



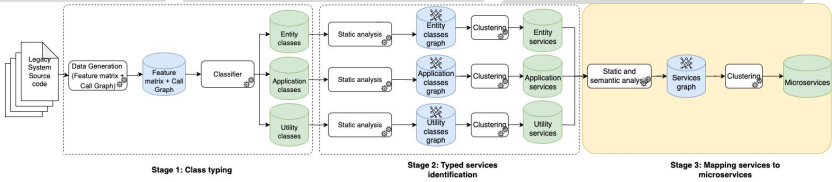


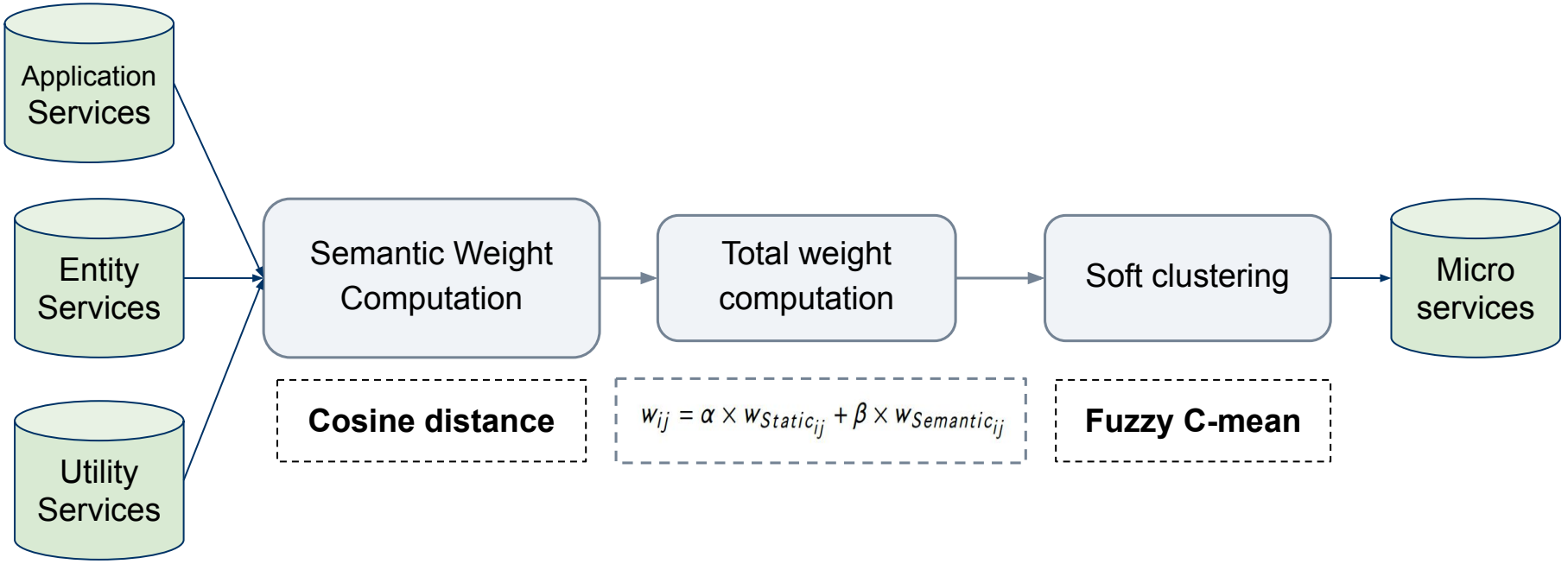
Stage 2: Typed Service identification





Stage 3: Microservices Mapping





Evaluation

Dataset: Compiere, FXML-POS, PetClinic, JForum 3

Benchmarking: Service Cutter and Topic modelling-based approach

Quantitative Evaluation: Precision, Recall, and F-measure against ground truths

Qualitative Evaluation: Cohesion and Modularity

Quantitative Evaluation

Legacy system	Approach	Precision	Recall	F-measure
Compiere	*ServiceCutter*	5%	21.7%	8.1%
	Topic modelling-based	22.5%	29.3%	25.4%
	MicroMiner	75.2%	72.8%	73.9%
FXML-POS	*ServiceCutter*	7.8%	33.3%	12.6%
	Topic modelling-based	29.4%	55.5%	38.4%
	MicroMiner	72%	88%	80%
PetClinic	*ServiceCutter*	8.5%	42%	14.2%
	Topic modelling-based	36.3%	57.1%	44.3%
	MicroMiner	71.4%	71.4%	71.4%
Jforum	*ServiceCutter*	4.7%	11.9%	6.7%
	Topic modelling-based	44.1%	45.2%	44.6%
	MicroMiner	54%	76.1%	63.1%

We obtained architecturally significant microservices have precision of **68.1%,** recall of **76.3%,** F-measure of **71.7%**

Our approach **outperformed** the two state of the art approaches

Qualitative Evaluation

Legacy system	Approach	IFN	CHM	CHD	SMQ	CMQ
Compiere	*ServiceCutter*	4.1	0.23	0.21	-0.17	-0.21
	Topic modelling-based	**2.1**	0.46	**0.54**	-0.02	0.01
	MicroMiner	3.4	**0.73**	0.53	**0.11**	**0.09**
FXML-POS	*ServiceCutter*	2.9	0.54	0.33	-0.12	0.01
	Topic modelling-based	2.5	0.47	**0.79**	-0.03	0.03
	MicroMiner	**2.3**	**0.69**	0.72	**0.09**	**0.05**
PetClinic	*ServiceCutter*	2.3	0.42	0.53	0.02	-0.03
	Topic modelling-based	2.1	0.43	**0.72**	0.03	0.02
	MicroMiner	**1.9**	**0.75**	0.67	**0.05**	**0.03**
Jforum	*ServiceCutter*	3.1	0.34	0.24	-0.01	-0.03
	Topic modelling-based	**2.3**	0.43	**0.62**	0.03	0.05
	MicroMiner	2.8	**0.67**	0.56	**0.04**	**0.O6**

IFN: MicroMiner results have better boundary definition

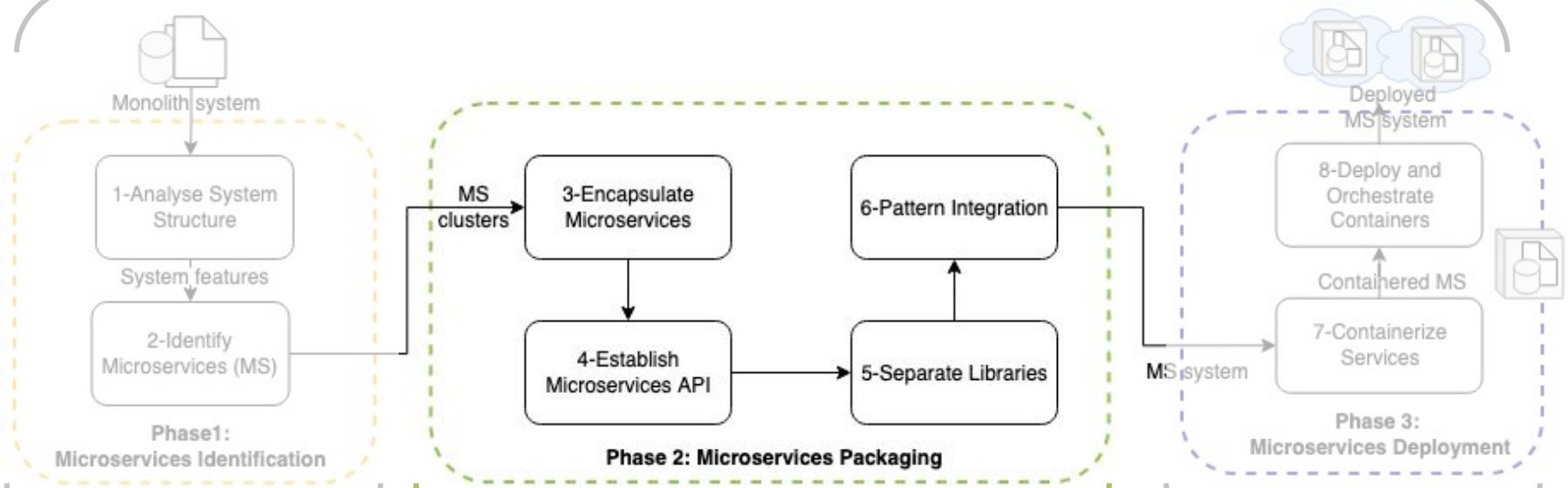
CHM, SMQ, CMQ: MicroMiner results have stronger functional cohesion and modularity.

Our approach **outperformed** the two state of the art approaches in most cases

Discussion and Threats

- Identifies utility, entity, and application services and merges them into coherent microservices
- Combines static and semantic relationships to align with domain boundaries
- Results depend on some parameters such as service size, and weight balancing
- The ML classifiers are trained on only four systems, results may vary in unseen domains

C1 A SLR of ML-based migration approaches



C2 MicroMiner: ML-based, fully automated identification approach

C3 MicroPacker: LLM-based, Automated Packaging Approach

C4 MiDKo: LLM-based, Fully automated deployment manifests generation

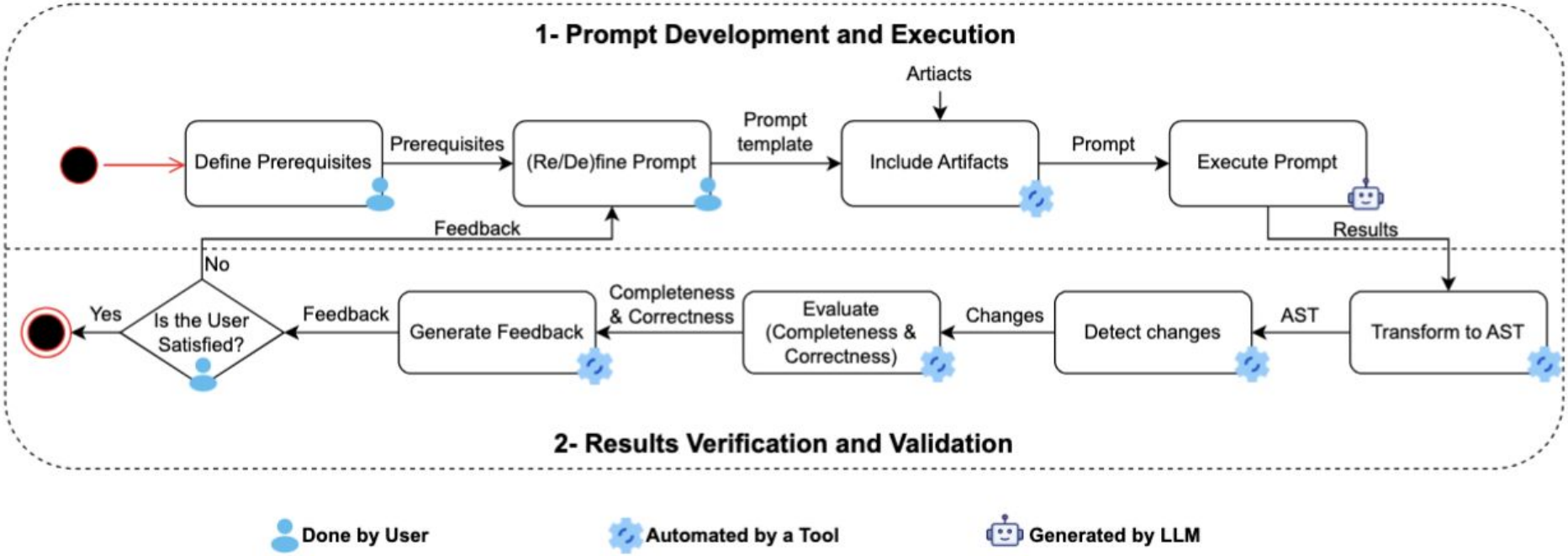
Context

- Microservice packaging demands extensive code generation and refactoring
- The packaging phase is largely overlooked in existing migration approaches
- Lack of automation and tooling

MicroPacker Overview

- An LLM-based approach for MS packaging for Automatic code generation
- Uses an iterative refinement process with a feedback loop
- Provide an automated tool

MicroPacker Approach



Evaluation

Case studies: FXML-POS and PetClinic

Quantitative Evaluation: Completeness, Correctness, and Operationality

Qualitative Evaluation: Manual Inspection

Quantitative Evaluation

$$\text{Effort Coverage Rate (\%)} = \left(1 - \frac{\text{\# Manual Lines (Added + Changed)}}{\text{\# Total Lines in System}}\right) \times 100$$

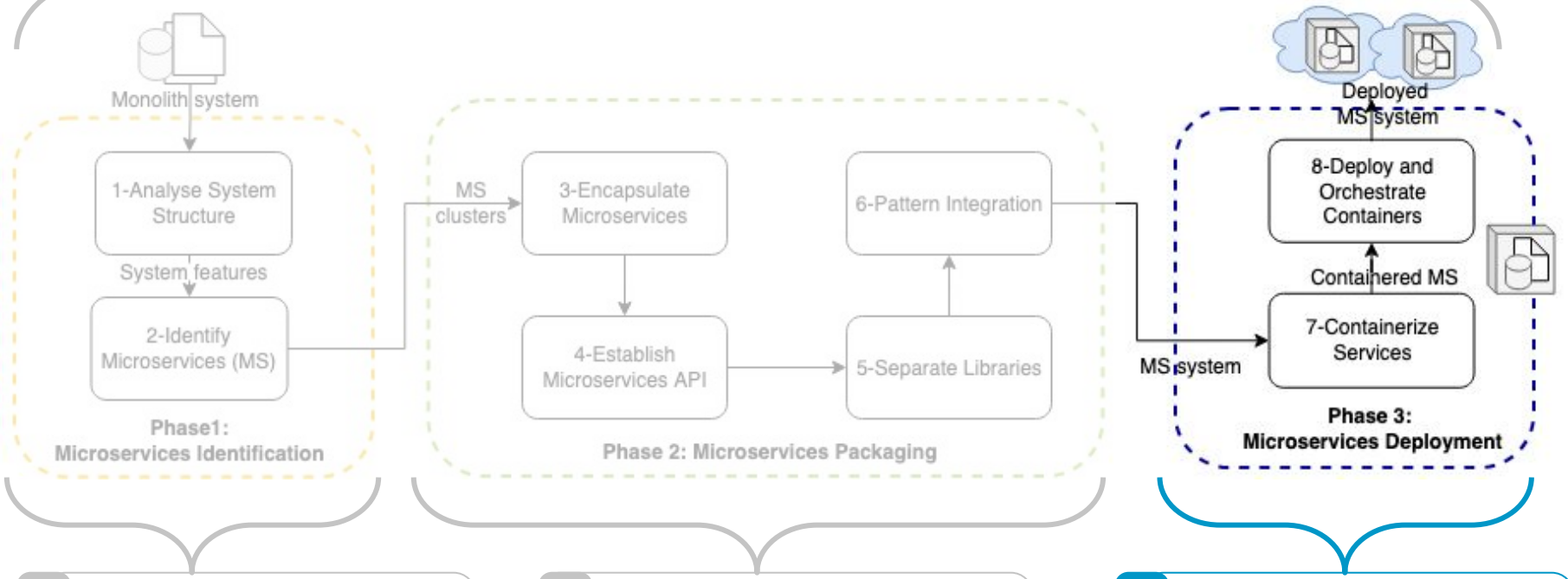
System	Total Lines	Effort Coverage Rate
PetClinic	423	86%
FXML-POS	630	82%

MicroPacker can significantly cut down manual effort in the packaging phase

Discussion and Threats

- Automates complex packaging tasks
- Consistent performance on PetClinic and FXML-POS despite structural differences
- Runtime behavior, maintainability, and performance not assessed
- Not evaluated on non-Java systems or industrial settings

C1 A SLR of ML-based migration approaches



C2 MicroMiner: ML-based, fully automated identification approach

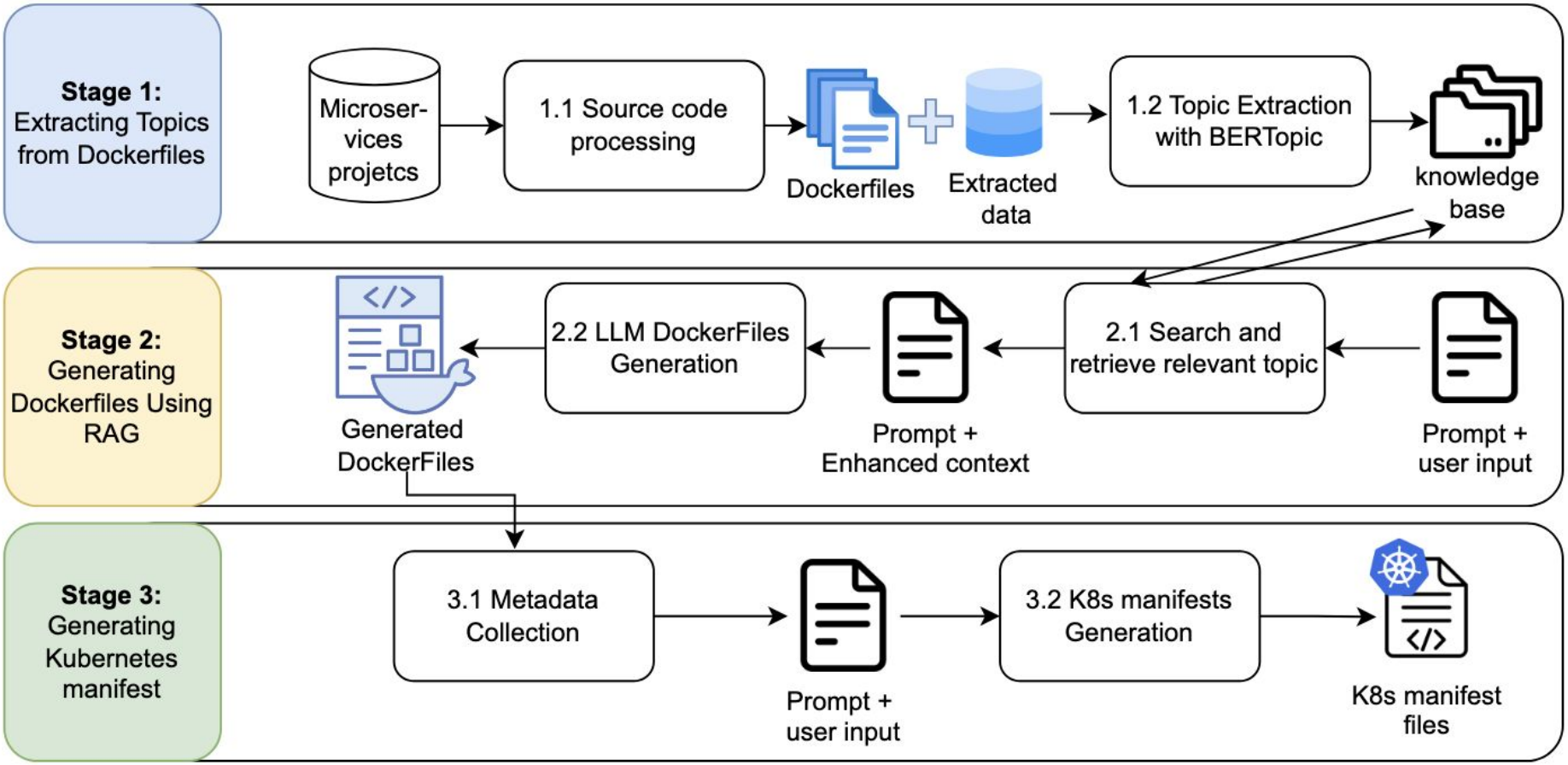
C3 MicroPacker: LLM-based, automated packaging approach

C4 MiDKo: LLM-based, Fully Automated Deployment Manifests Generation

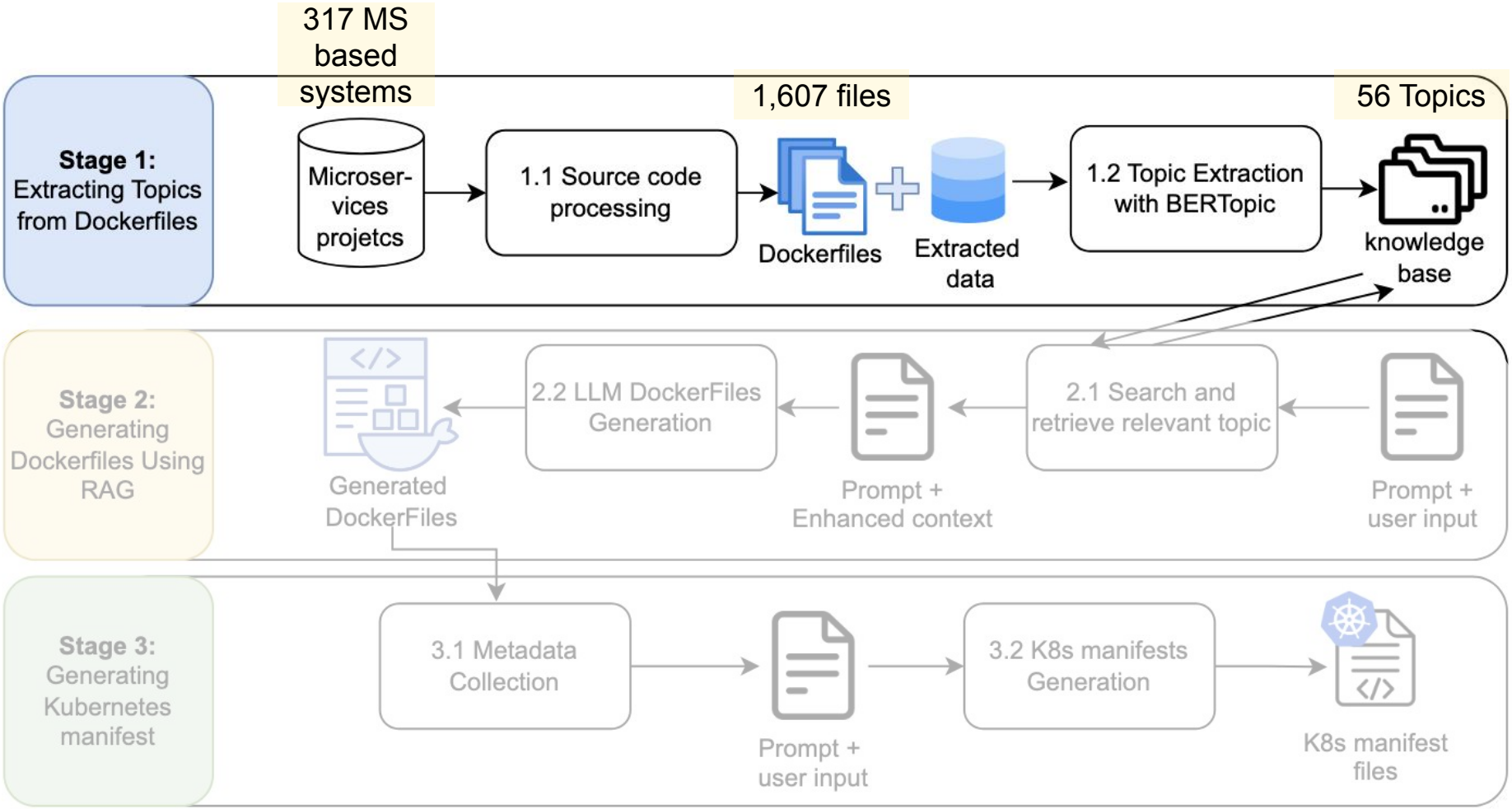
Context

- Containerisation has become standard practice for microservices deployment
- Most existing works focus on resource allocation, autoscaling, and failure detection
- Existing tools (e.g., Helm, Kustomize) rely on static templates with limited adaptability

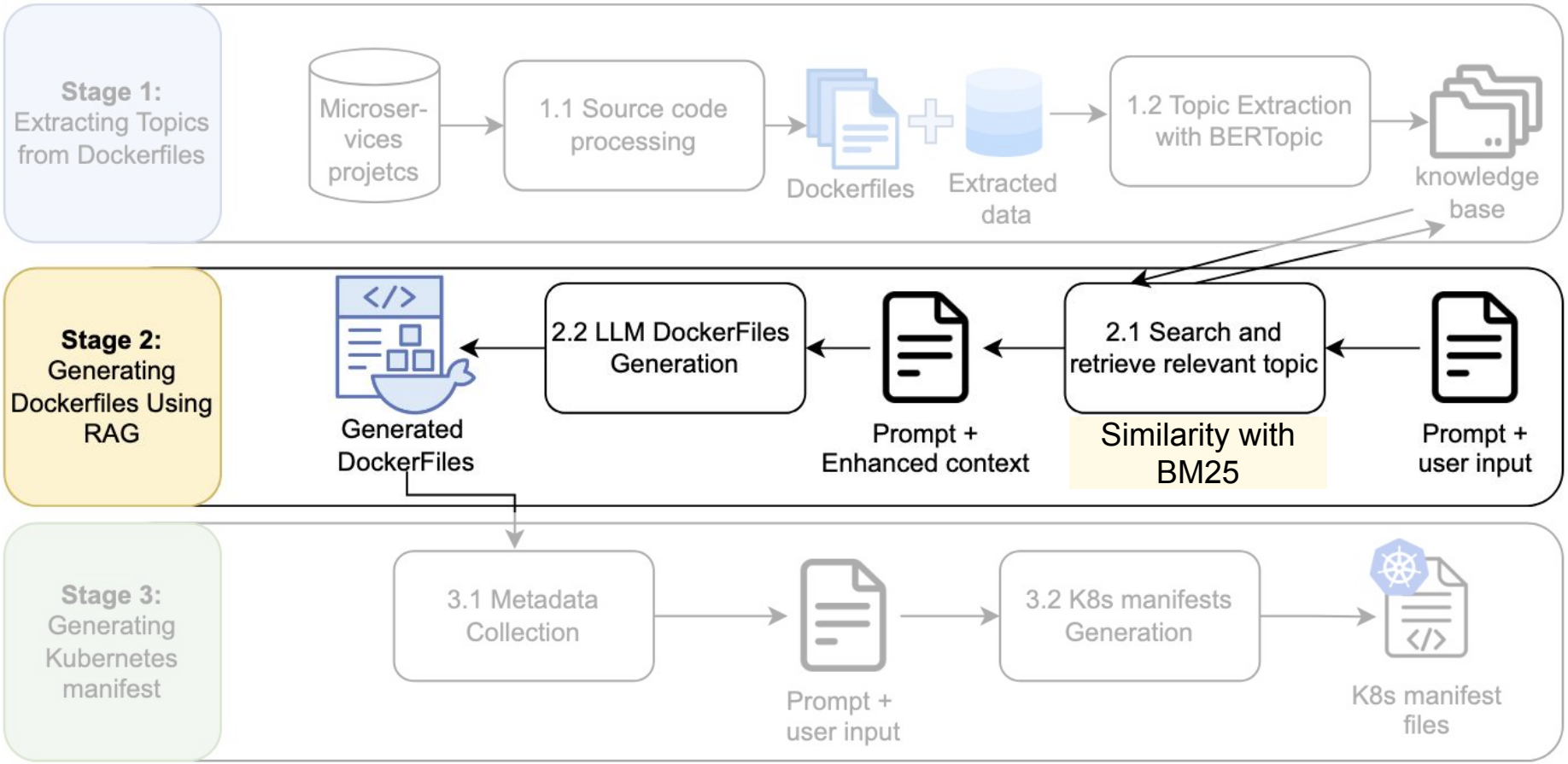
MiDKo Overview



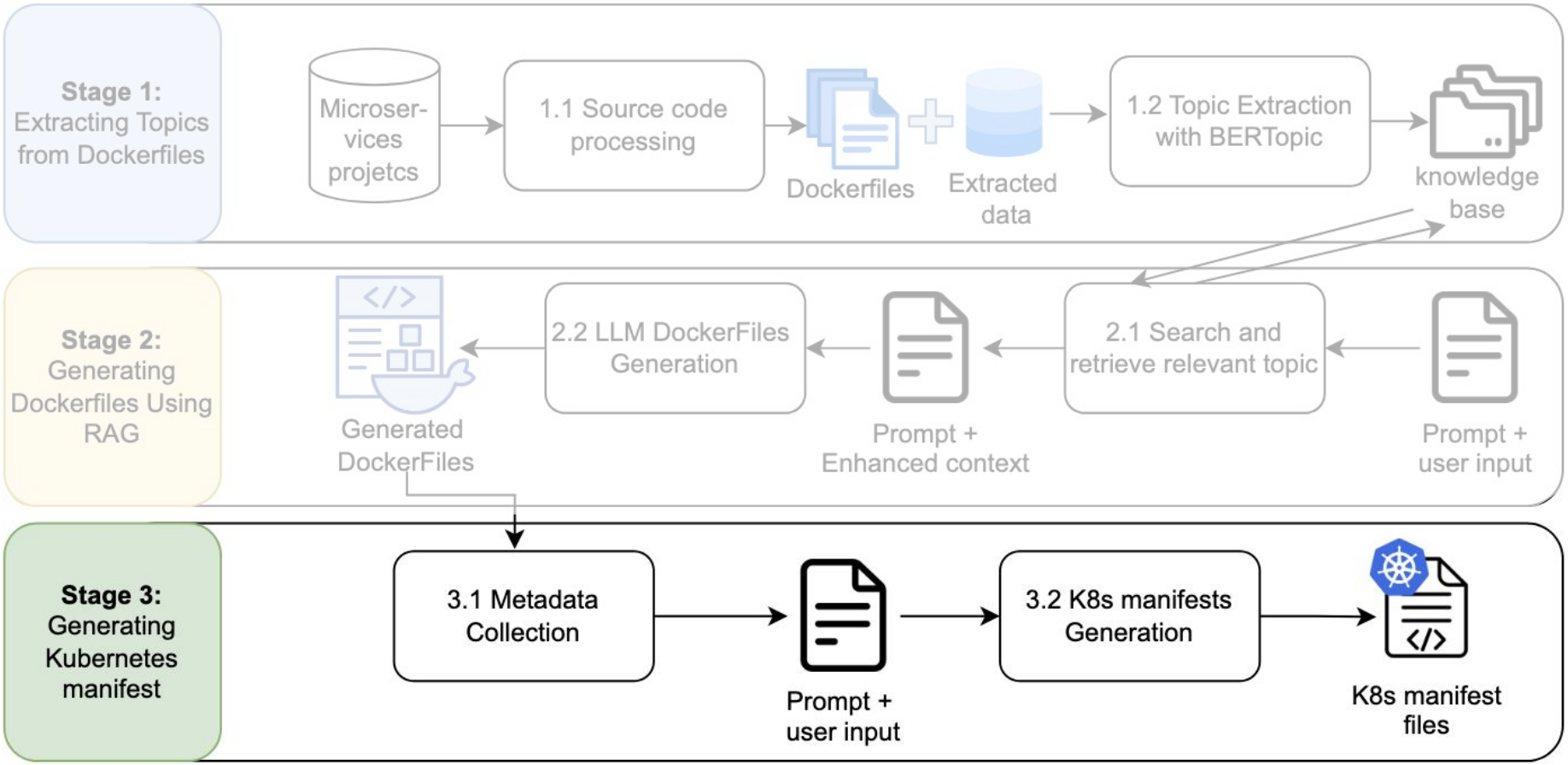
MiDKo Overview



MiDKo Overview



MiDKo Overview



Evaluation

Dataset: 317 open source projects

Benchmarking: Ground truth, MiDKo-without-Topics

Quantitative Evaluation: Syntactic Correctness and Similarity Analysis

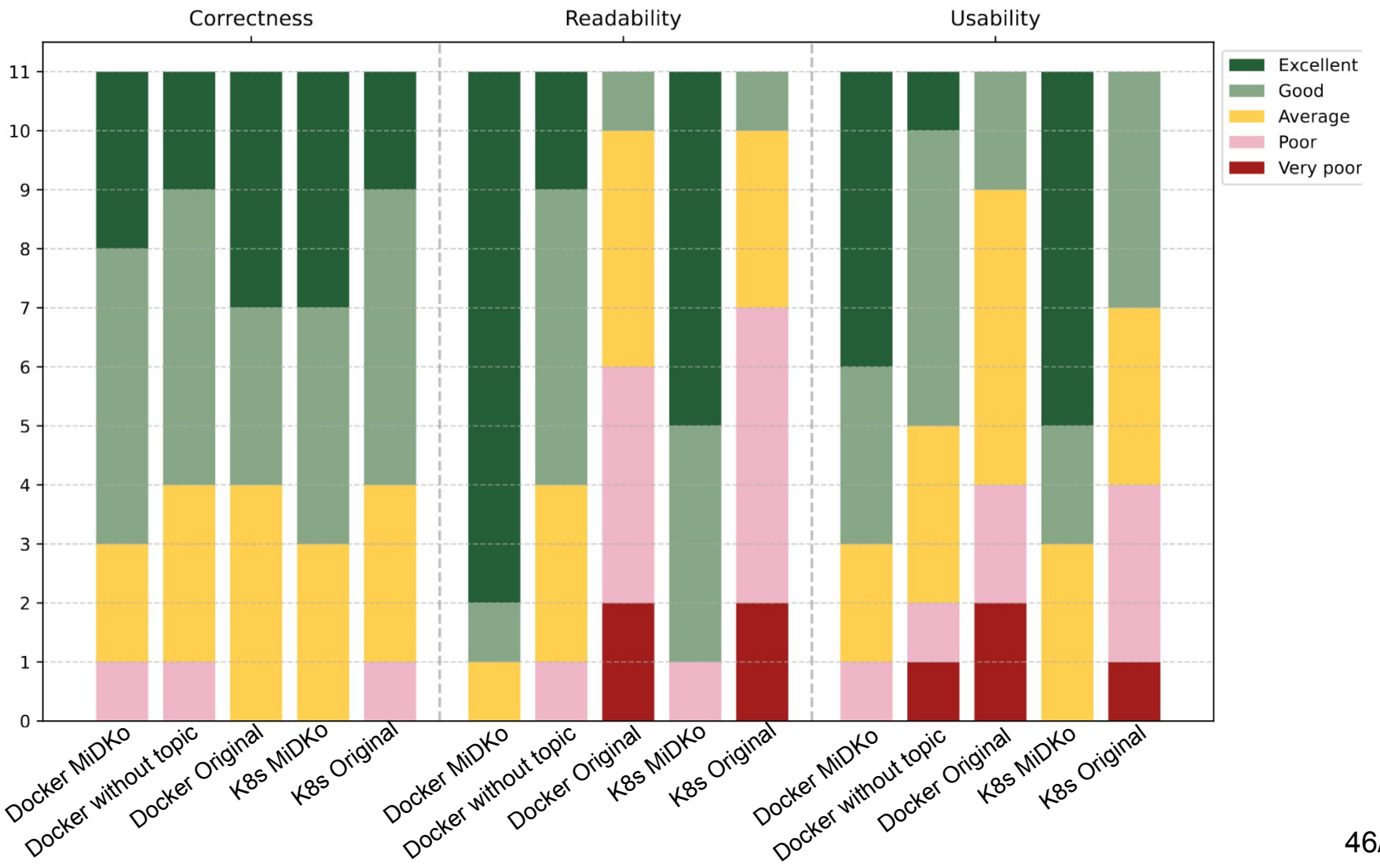
Qualitative Evaluation: Survey

Quantitative Evaluation

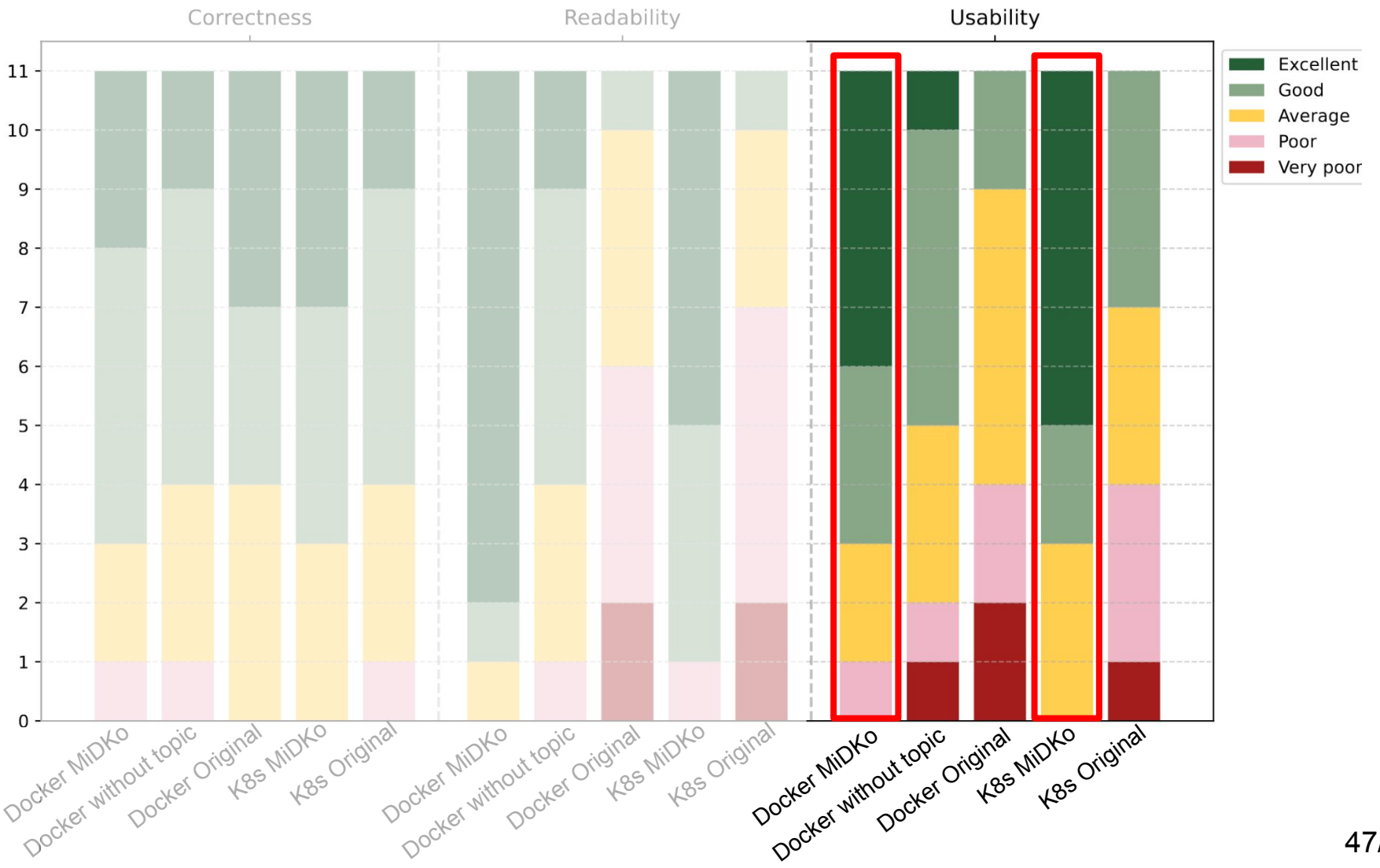
Linters: Hadolint and Kube-score

Approach	Syntax Correctness (#files)	
	Clean ↑	Corrupted ↓
Docker_original	1,203 (74%)	404
Docker_without_topics	1,448 (90%)	159
Docker_MiDKo	1,459 (91%)	148
K8s_original	72 (46%)	83
K8s_MiDKo	109 (70%)	46

Qualitative Evaluation



Qualitative Evaluation



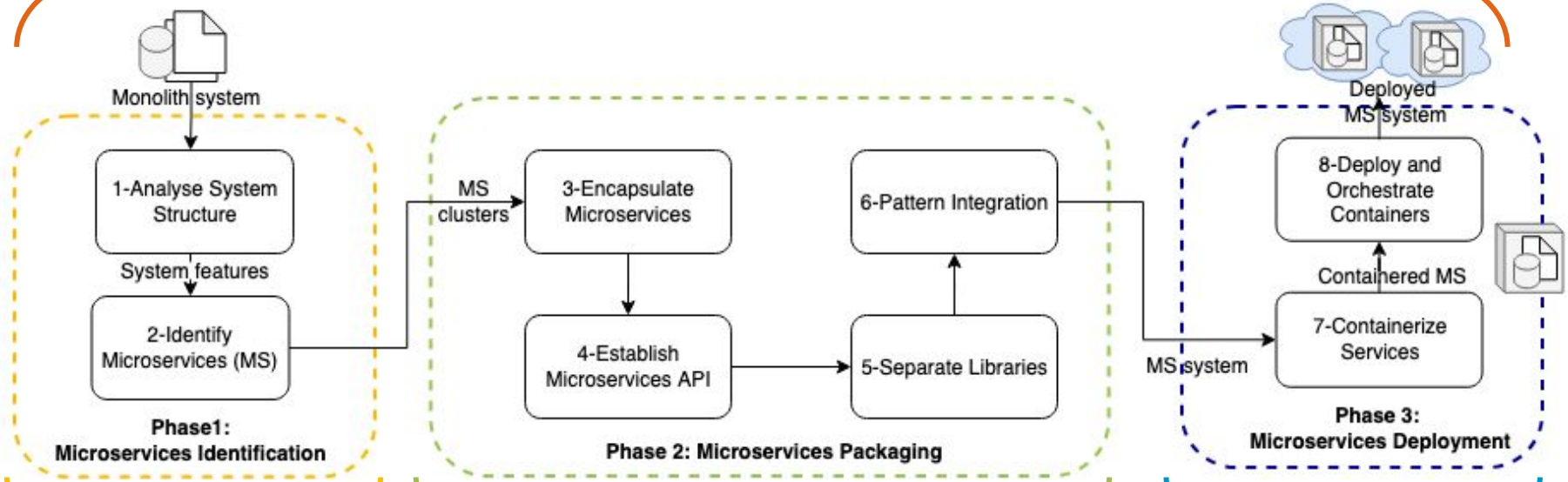
Discussion and Threats

- MiDKo-generated Dockerfiles/K8s manifests have high Syntactic Correctness
- Produces more usable and readable configs than many open-source ones
- No support for multi-stage builds
- Require some input from the user

Conclusion

Conclusion

C1 A SLR of ML-based Migration Approaches



C2 MicroMiner: ML-based, Fully Automated Identification Approach

C3 MicroPacker: LLM-based, Automated Packaging Approach

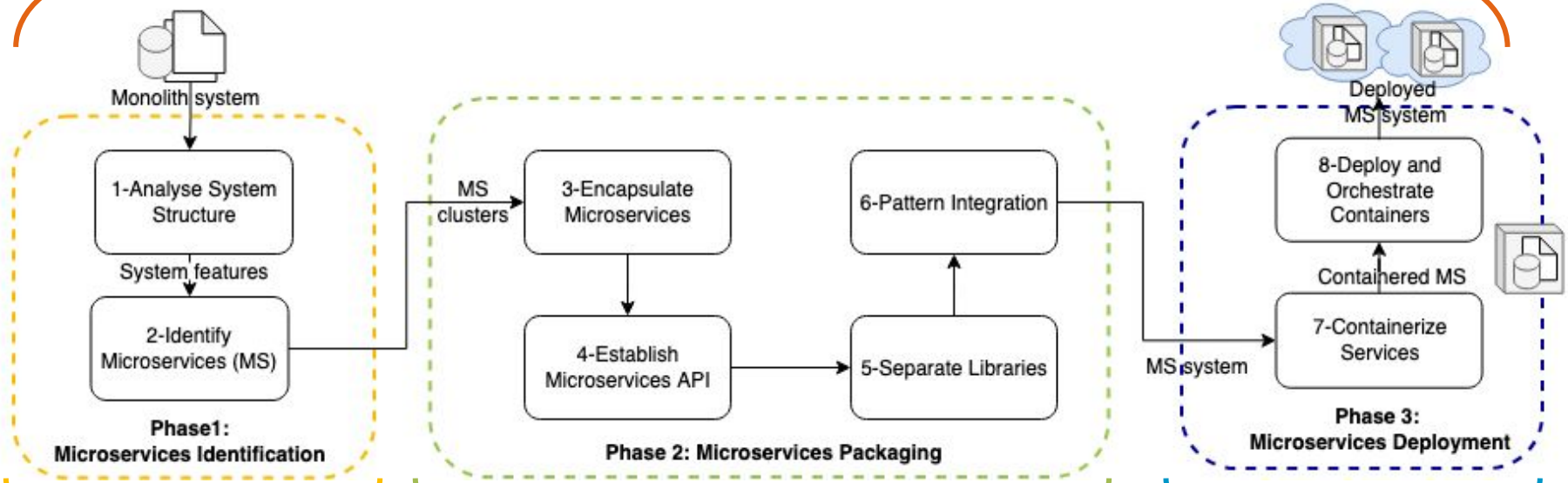
C4 MiDKo: LLM-based, Fully Automated Deployment Manifests Generation

Conclusion

The migration from monolithic systems to microservices can be **automated**, to minimise manual effort and while ensuring accuracy, by **leveraging machine learning** and **large language models** across the three phases: **identification, packaging, and deployment.**

Conclusion

C1 A SLR of ML-based Migration Approaches



C2 MicroMiner: ML-based, Fully Automated Identification Approach

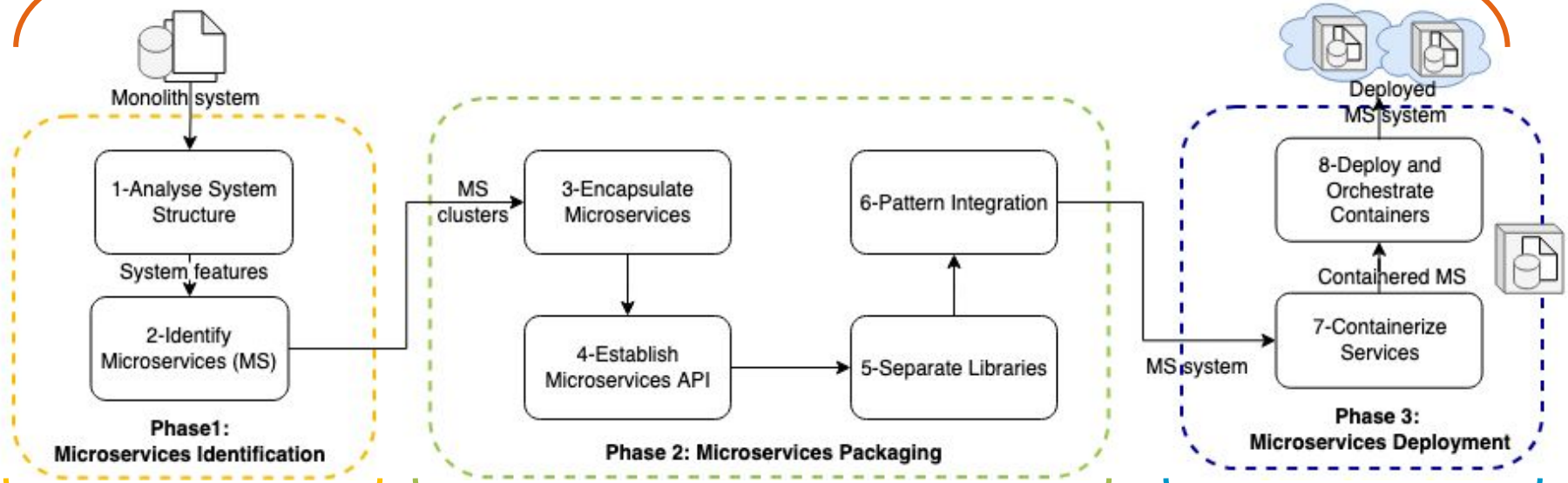
C3 MicroPacker: LLM-based, Automated Packaging Approach

C4 MiDKo: LLM-based, Fully Automated Deployment Manifests Generation

Validation on industrial setting

Conclusion

C1 A SLR of ML-based Migration Approaches



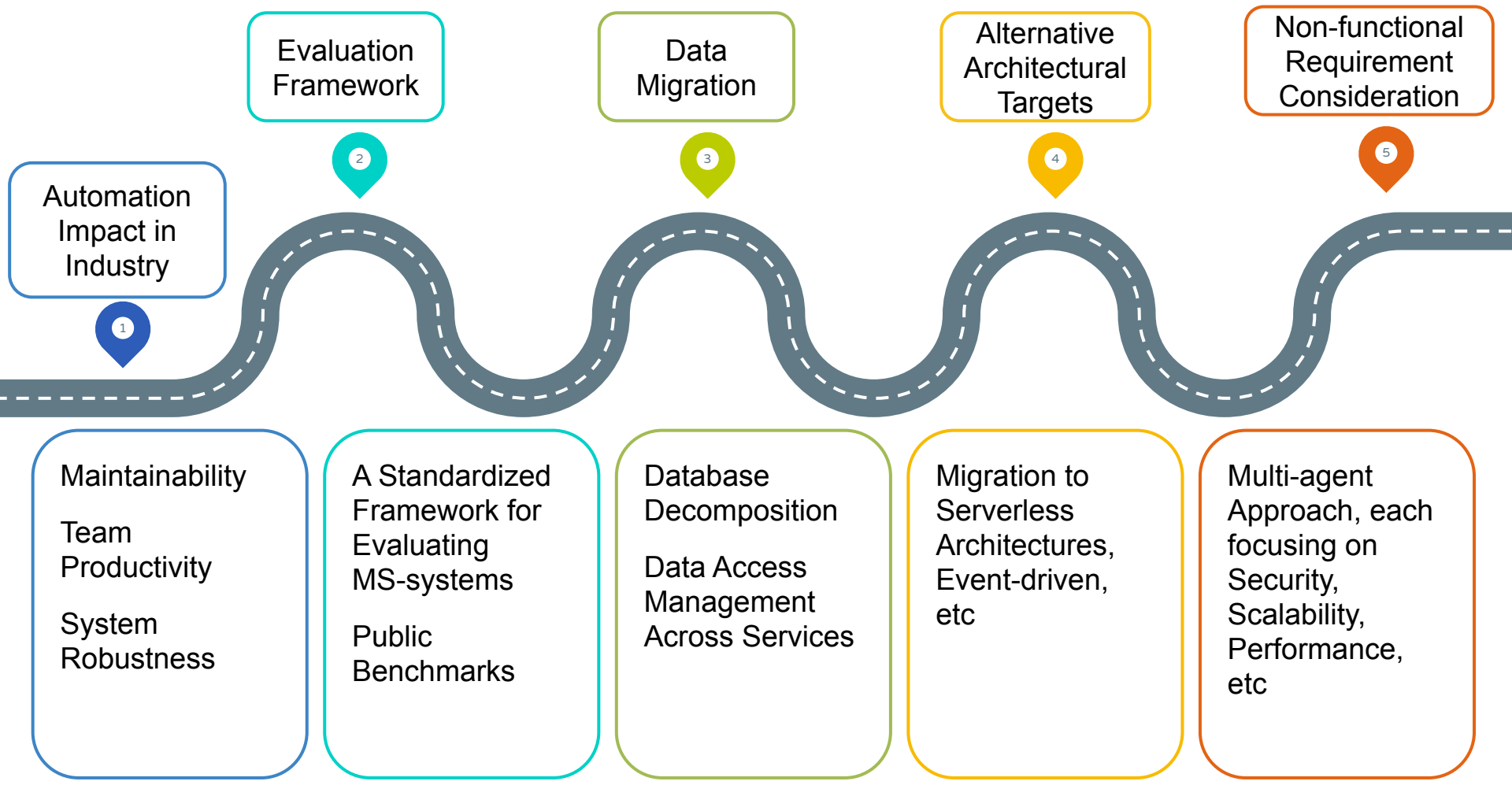
C2 MicroMiner: ML-based, Fully Automated Identification Approach

C3 MicroPacker: LLM-based, Automated Packaging Approach

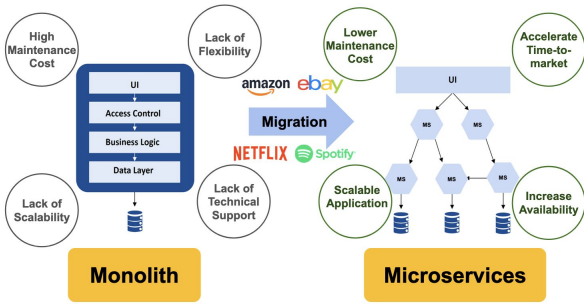
C4 MiDKo: LLM-based, Fully Automated Deployment Manifests Generation

Data Migration

Future Works



Monoliths to Microservices

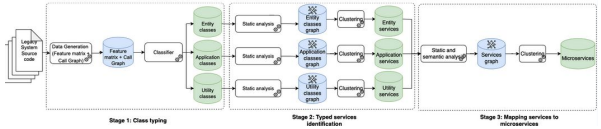


SLR of ML-based Migration approaches

RQ1: Which migration **phases** are automated by ML?
RQ2: How are **inputs** used by ML migration approaches?
RQ3: What **ML approaches** do researchers apply?
RQ4: How do researchers **evaluate** ML approaches?

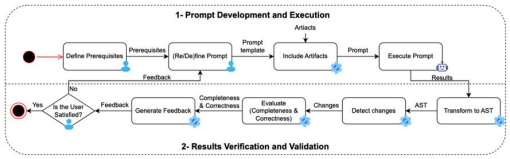
MicroMiner Overview

- ✔ Based on semantic and static analysis
  - Respecting domain principals
- ✔ Based on typed services
  - Insuring complete architectures layers
- ✔ Provide an automated tool

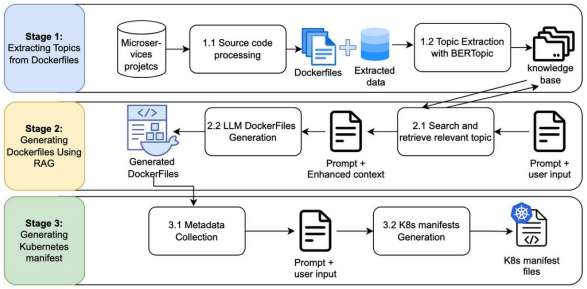


MicroPacker Overview

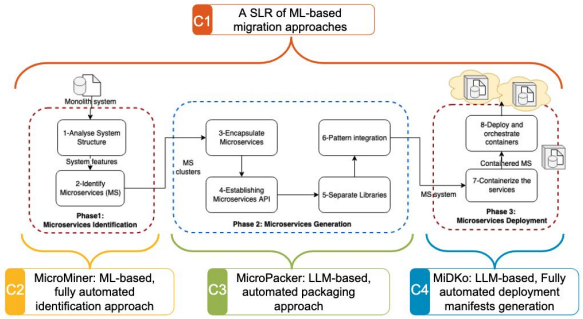
- ✔ An LLM-based approach for MS packaging
  - Automate code generation
- ✔ Uses an iterative refinement process with a feedback loop
- ✔ Provide an automated tool



MiDKo Overview



Conclusion



References

[1] *Jonas Fritzsch, Justus Bogner, Stefan Wagner, and Alfred Zimmermann.* ***Microservices migration in industry: intentions, strategies, and challenges****. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), pages 481–490. IEEE, 2019.*

[2] Trabelsi Imen, Manel Abdellatif, Abdalgader Abubaker, Naouel Moha, Sébastien Mosser, Samira Ebrahimi-Kahou, and Yann-Gaël Guéhéneuc. ***"From legacy to microservices: A type-based approach for microservices identification using machine learning and semantic analysis."*** **J**ournal of Software: Evolution and Process (2022)

[3] Trabelsi Imen, Popa Bianca, Péreyrol Jérémie, Beaulieu Pier-Olivier, and Moha Naouel. ***"MicroMatic: Fully Automated Microservices Identification Approach From Monolithic Systems."*** SERP4IoT'24 ICSE workshop (2024).

[4] Trabelsi Imen, Moha Naouel, Guéhéneuc Yann-Gaël, and Geffard Lucas. ***"MAGNET: Method-based Approach using Graph Neural Network for Microservices Identification."*** 21st IEEE International Conference on Software Architecture (ICSA 2024).

[5] Pedreira, Oscar, et al. **"Gamification in software engineering–A systematic mapping**." *Information and software technology* 57 (2015): 157-168.